小米便签开源软件的维护方案及成果

撰写人：马坚、葛宏波、肖锟钰、赵升

1．开源软件简介

Mi Note 便签是小米便签的社区开源版，由 MIUI 团队(www.miui.com) 发起并贡献第一 批代码，是较为流行的记事软件。代码总计 8.7K 行，主要使用 JAVA 语言编写，在 Android 平台上进行开发维护，涉及到 Android 编程的各方面。

2．软件维护需求描述

2.1 开源软件分析

小米便签作为一个事件提醒软件，已经具备了一个便签应当有的所有功能，一定意义上来说已经是一个比较完备的软件，但在细节上仍有加以改进的地方。如用 SonarQube 扫描后仍有不少的 Bug，其中包含一些可能导致便签崩溃的主要、重要程度的缺陷。在个人隐私方面有所欠缺：便签没有加密功能；用户体验度方面除了设置闹钟还可以添加置顶、设置星标等功能；视觉上相对单一，可以增加背景库及欢迎界面来提高便签视觉上的冲击；现在文本输入功能都会有计数，这是小米便签遗漏的地方，以上都是可以维护的切入点。

2.2 开源软件维护需求

1. 新增功能描述

（1）载入欢迎界面：打开小米便签时，会有短暂的欢迎界面，2s 后界面自动消失，此时进入小米便签主界面。

（2）更换便签背景：小米便签源代码中的背景太单调，增加了背景库供用户选择。

（3）统计输入字符数：在小米便签编辑界面，统计当前已经输入了多少字符，方便用户知道自己写了多少字。

（4）设置星标便签：用户将重要的便签设置成星标便签不可被删除，同时在 list 表中用五角星进行标识，这样可以防止批量删除或者删除文件夹时误删重要的便签。

（5）实现便签置顶：置顶显示是目前主流的操作，由于小米便签开发比较早并没有实现这个功能，为了提高用户体验度，增加置顶功能使得用户可以将那些需要经常看到的便签置顶在便签列表上方，防止遗忘。

（6）实现九宫格手势加密：有一些便签中的内容需要被加密保护，这样就需要给便签增加一个锁来确保安全，加密的方法有很多这里选择用九宫格手势加密。但由于九宫格加密前人已经实现了，这里主要是在前人的基础上进行细节的完善，如删除密码时进行密码验证、忘记密码时可进行密码找回、优化界面切换时的使用友好度。

（7）实现云端存储：小米便签实现连接阿里云的 RDS，将本地的便签封装成 JSON 消息包发送到云端数据库进行存储。本地的小米便签也可以通过登陆远端数据库进行同步拉取。

2. 代码缺陷描述

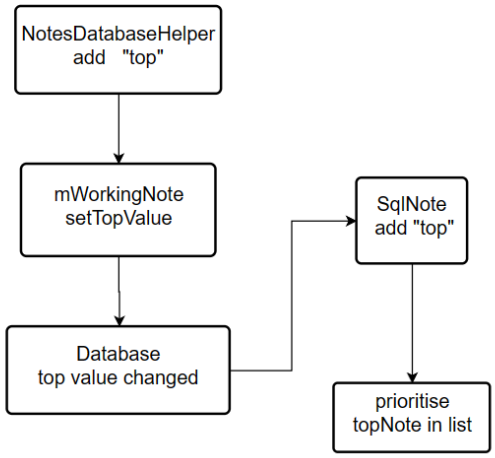
主要为空指针和语法错误，详细情况见 3.4 sonarqube 分析

3．维护设计和实现方案

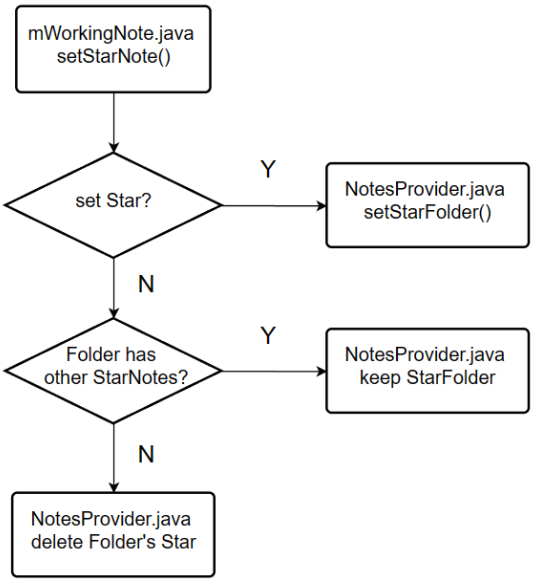
3.1 开源软件体系结构的设计及调整

3.2 维护需求的实现方案

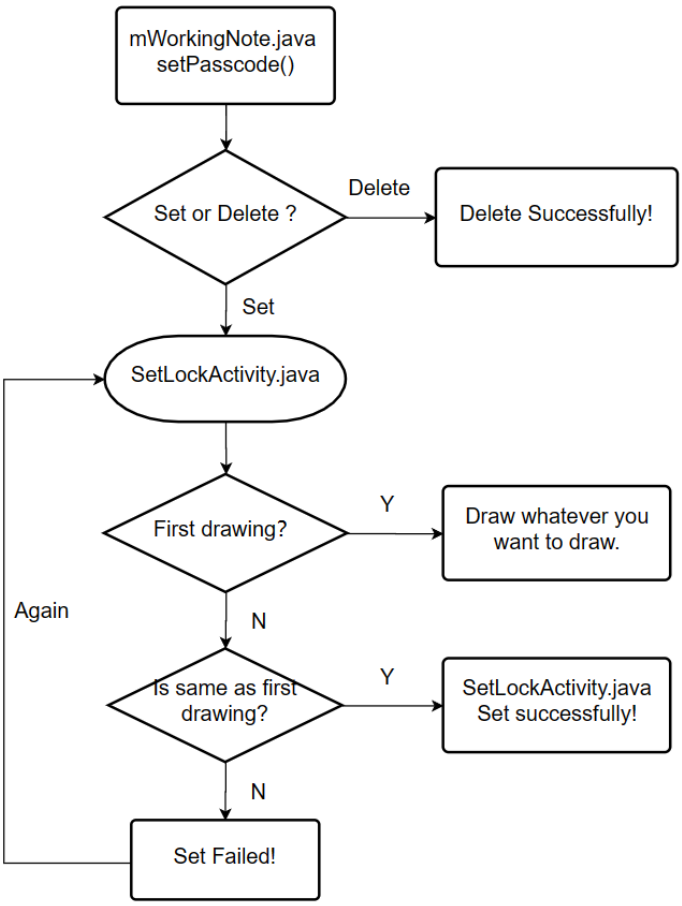
1. 置顶便签



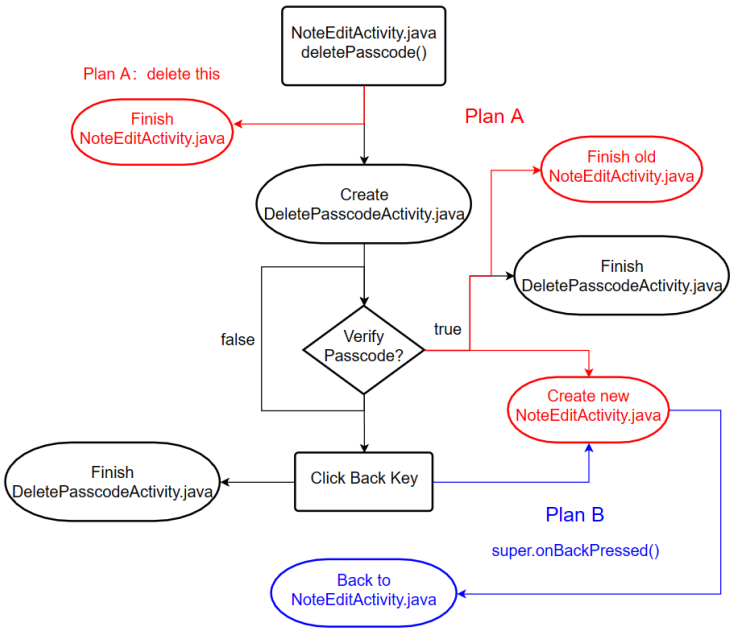
2. 设置星标



3. 加密



九宫格加密中主要进行了细节维护：删除密码时先进行密码验证、设置与删除密码时 Activity 的切换。其中删除密码时先进行密码验证与进入便签时先验证密码异曲同工，下面是 Activity 切换的流程图



3.3 维护需求的类设计

- 载入欢迎界面：

新增活动 SplashActivity，当打开小米便签时，首先进行此活动，这时会有 2s 中的欢迎界面载入。重写方法 run 使得在规定时间到达时自动结束当前活动（即欢迎界面）。

代码如下：

```
package net.micode.notes.ui;

import android.content.Intent;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import net.micode.notes.R;

public class SplashActivity extends Activity {
    Handler mHandler=new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); //加载启动界面
        setContentView(R.layout.activity_splash); //加载启动图片

        // 当计时结束时，跳转至NotesListActivity
        mHandler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent=new Intent();
                intent.setClass(SplashActivity.this, NotesListActivity.class);
                startActivity(intent);
                finish(); //销毁欢迎页面
            }
        }, 2000); // 2 秒后跳转
    }
}
```

- 设置背景图片：

小米便签的背景过于单调，为了增添便签的多样性，添加了背景切换功能：

首先在 preference.xml 文件中添加布局：

```
<PreferenceCategory
        android:key="pref_key_bg_image_appear">
</PreferenceCategory>
```

在设置界面（NotePreferenceActivity）中获取布局，并在 onCreate 方法中初始化：

```
private void loadBgPicsPreference(){
    mBgPicsCategory.removeAll();

    Preference bgpicsPref = new Preference(this);
    bgpicsPref.setTitle("设置背景图片");
    bgpicsPref.setOnPreferenceClickListener(new OnPreferenceClickListener() {
        @Override
        public boolean onPreferenceClick(Preference preference) {
            Intent intent = new Intent(NotesPreferenceActivity.this,
                    SelectBgPicsActivity.class);
            startActivity(intent);
            return true;
        }
    });

    mBgPicsCategory.addPreference(bgpicsPref);
}
```

这时点击设置是就会多出一下选项：“设置背景图片”，现在为这个选项进行具体功能的实现。新建一个选择背景图片的活动 SelectBgPicsActivity，它的页面布局很简单，就是一个 ImageView 用于存放背景图片和 Button 应用用于选择确认背景图片：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center|top"
    android:orientation="vertical"
    tools:context="net.micode.notes.ui.SelectBgPicsActivity"
    android:fitsSystemWindows="true">

    <ImageView
        android:id="@+id/bg_images"
        android:layout_width="430dp"
        android:layout_height="656dp"
        android:src="@drawable/water">
    </ImageView>

    <Button
        android:id="@+id/apply_pic"
        android:layout_width="match_parent"
        android:layout_height="51dp"
        android:layout_alignParentBottom="true"
        android:layout_gravity="bottom"
```

```
        android:text="应用" />

</LinearLayout>
```

回到活动中来，定义 int 类型数组存放背景图片的 id 值：

```
private int[] BgPicsId = new int[]{
    R.drawable.water,
        R.drawable.grassland,
};
```

通过监听手势移动进行图片切换：

```
//当Activity 被触摸时回调
public boolean onTouchEvent(MotionEvent event){
    return gestureDetector.onTouchEvent(event);
}
//自定义GestureDetector 的手势识别监听器
private GestureDetector.OnGestureListener onGestureListener
        = new GestureDetector.SimpleOnGestureListener(){
    //当识别的手势是滑动手势时回调onFinger 方法
    public boolean onFling(MotionEvent e1,MotionEvent e2,float velocityX,float
velocityY){
        //得到手触碰位置的起始点和结束点坐标 x , y , 并进行计算
        float x = e2.getX()-e1.getX();
        float y = e2.getY()-e1.getY();
        //通过计算判断是向左还是向右滑动
        if(x > 0){
            count++;
            count%=(BgPicsId.length);        //想显示多少图片，就把定义图片的数组长度-1
        }else if(x < 0){
            count--;
            count=(count+(BgPicsId.length))%(BgPicsId.length);
        }

        mBgPics.setImageResource(BgPicsId[count]);  //切换 imageView 的图片
        return true;
    }
};
```

这样左滑右滑就可以纵享丝滑地切换库中背景图片进行预览了。

这是监听应用按钮：点击时将 NotesListActivity 中的背景图片设置成当前预览的图片：

```
//监听应用设置按钮
public void onClick(View v){
    switch (v.getId()){
        case R.id.apply_pic:
            NotesListActivity.getBgPicId(BgPicsId[count]);
            NotesPreferenceActivity activity = new NotesPreferenceActivity();
            activity.NotesPreferenceActivity.finish();
```

```
            finish();
            break;
        default:
            break;
    }
}
```

- 统计输入字符：

  在便签编辑界面，通过顶端任务栏展示当前编辑框内已输入内容的字符数，这对用户来说十分地友好，其实现的思路如下：获取文本框内容，将内容中的空格换行符去除转化成 String 类型字符串，统计字符串长度并显示：

  定义函数将 text 类型文本转化成 String 类型字符串：

```
private String Textchange(String oriText){
    StringBuffer stringBuffer = new StringBuffer(oriText);
    int Flag1 = -1;
    int Flag2 = -1;
    do {//不计入表示图片的字符
        Flag1 = stringBuffer.indexOf("<img");
        Flag2 = stringBuffer.indexOf(">");
        if (Flag1 != -1 && Flag2 != -1) {
            stringBuffer = stringBuffer.replace(Flag1, Flag2+1, "");
        }
    } while (Flag1 != -1 && Flag2 != -1);

    do {//不计入换行字符
        Flag1 = stringBuffer.indexOf("\n");

        if (Flag1 != -1){
            stringBuffer = stringBuffer.replace(Flag1, Flag1+1, "");
        }
    } while (Flag1 != -1);
    do {//不计入空格字符
        Flag1 = stringBuffer.indexOf(" ");

        if (Flag1 != -1) {
            stringBuffer = stringBuffer.replace(Flag1, Flag1+1, "");
        }
    } while (Flag1 != -1);
    return stringBuffer.toString();
}
```

  统计字符串长度并展示：

```
private void count() {
    editText = (EditText) findViewById(R.id.note_edit_view);
```

```
    textView = (TextView) findViewById(R.id.text_num);
    mNoteEditor.addTextChangedListener(new TextWatcher() {
        int currentLength = 0;
        @Override
        public void beforeTextChanged(CharSequence s,
                                      int start, int count, int after) {
            textView.setText("字数：" + currentLength);
        }

        @Override
        public void onTextChanged(CharSequence s,
                                  int start, int before, int count) {
            currentLength = Textchange(mNoteEditor.getText().toString()).length();
        }

        @Override
        public void afterTextChanged(Editable s) {
            textView.setText("字数：" + currentLength);
        }
    });
}
```

这样就可以在便签编辑界面一目了然地看到已经输入了多少字符了。

- 置顶与星标便签

Data 包中增添属性：top、star、passcode,

```
public static final String TOP = "top";
public static final String STAR = "star";
public static final String PASSCODE = "passcode";
```

同时在 NotesDatabaseHelper 中进行更新：

```
if (oldVersion == 4) {
    upgradeToV5(db);
    oldVersion++;
}
if (oldVersion == 5) {
    upgradeToV6(db);
    oldVersion++;
}

private void upgradeToV5(SQLiteDatabase db) {
    db.execSQL("ALTER TABLE " + TABLE.NOTE + " ADD COLUMN " + NoteColumns.TOP
            + " INTEGER NOT NULL DEFAULT 0");
}
private void upgradeToV6(SQLiteDatabase db) {
```

```
        db.execSQL("ALTER TABLE " + TABLE.NOTE + " ADD COLUMN " + NoteColumns.STAR
                + " INTEGER NOT NULL DEFAULT 0");
    }
```

并且修改

mWorkingNote 中新增方法 setTop 和 setStar:

```
public void setTop(String Top){
    if (!mTop.equals(Top)) {
        mTop = Top;
        mNote.setTopValue(NoteColumns.TOP,mTop);
    }
    if (mNoteSettingStatusListener != null) {
        mNoteSettingStatusListener.onTopChanged(Top);
    }
}

public void setStar(String Star){
    if (!mStar.equals(Star)){
        mStar = Star;
        mNote.setStarValue(NoteColumns.STAR,mStar);
        NotesProvider.setStarFolder(mFolderId,
                NotesProvider.isStarFolder(mNoteId, mFolderId,Star.equals("1")));
        if (mNoteSettingStatusListener != null) {
            mNoteSettingStatusListener.onStarChanged(Star);
        }
    }
}
```

上述两个方法中用到了新方法 setTopValue、setStarValue，setStarFolder 这三个新方法用来在数据库中修改 top 与 star 的值，他的代码如下：

```
public void setTopValue(String key, String value) {
    mNoteDiffValues.put(key, value);
}
public void setStarValue(String key, String value) {
    mNoteDiffValues.put(key, value);
}
public static void setStarFolder(long FolderId, boolean isStarFolder){
    SQLiteDatabase db;
    db = mHelper.getWritableDatabase();
    ContentValues values = new ContentValues();
    if(isStarFolder){
        values.put(NoteColumns.STAR,"1");
    } else{
        values.put(NoteColumns.STAR,"0");
```

```
    }
    String condition = NoteColumns.ID + " = ?";
    String[] args = {String.valueOf(FolderId)};
    db.update(NotesDatabaseHelper.TABLE.NOTE, values, condition, args);
}
```

其中 setStarValue 是设置当前编辑便签的星标值；setStarFolder 是设置便签所在文件夹的星标值，这个方法是在 Data 包的 NoteProvider.java 文件中实现的：当便签内含有至少一条星标便签时，文件夹就会被设置成星标文件夹。判断是否需要设置星标文件夹的代码如下：

```
public static boolean isStarFolder(long NoteId, long FolderId, boolean setStar){
    if(setStar){
        return true;
    }
    SQLiteDatabase db;
    db = mHelper.getWritableDatabase();
    Cursor cursor = null;
    cursor = db.query(NotesDatabaseHelper.TABLE.NOTE, null, null, null, null, null,
null);
    //判断游标是否为空
    if (cursor.moveToFirst()) {
        //遍历游标
        for (int index = 1; index < cursor.getCount(); index++) {
            cursor.moveToNext();
            if (cursor.getLong(SqlNote.ID_COLUMN) == NoteId && !setStar){
                continue;
            }
            if (cursor.getLong(1) == FolderId){
                if (cursor.getString(SqlNote.STAR_COLUMN).equals("1")){
                    cursor.close();
                    return true;
                }
            }
        }
    }
    cursor.close();
    return false;
}
```

NoteEditActivity 中在菜单栏对于位置增加按钮链接：

```
case R.id.menu_set_top:
    mWorkingNote.setTop((mWorkingNote.getTopId())==1 ? "0" : "1");
    break;
case R.id.menu_cancel_top:
```

```
            mWorkingNote.setTop((mWorkingNote.getTopId())==0 ? "1" : "0");
            break;
        case R.id.menu_set_star:
            mWorkingNote.setStar(mWorkingNote.getStarId()==1? "0" : "1");
            System.out.println(mWorkingNote.hasPasscode());
            break;
```

因为每次只需要显示一个按钮，比如当我没有置顶时我要显示“置顶”按钮、已经置顶的便签要显示“取消置顶”按钮，通常有两种方法，一种是定义两个布局，在不同情况下选择性显示按钮，另一种方法是定义一个布局按钮，在不同情况下修改按钮的文本显示。两种不同方法的代码如下：

```
    if (mWorkingNote.getTopId() == 1 ) {
        menu.findItem(R.id.menu_set_top).setVisible(false);
    } else if(mWorkingNote.getTopId() == 0){
        menu.findItem(R.id.menu_cancel_top).setVisible(false);
    }
    if (mWorkingNote.getStarId() == 1){
        menu.findItem(R.id.menu_set_star).setTitle(R.string.menu_delete_star);
    }
```

增加按钮布局：

```
    <item
        android:id="@+id/menu_set_top"
        android:title="@string/menu_set_top"/>

    <item
        android:id="@+id/menu_cancel_top"
        android:title="@string/menu_cancel_top"/>

    <item
        android:id="@+id/menu_set_star"
        android:title="@string/menu_set_star"/>
```

上述代码的添加可以实现小米便签属性的添加并修改值，但是置顶便签还不能真正的置在顶处，星标便签仍然可以被删除，接下来就是具体功能内容的实现。

首先是置顶便签真正意义上置在顶处，思路是获取便签的属性，在便签列表中按照 top 值降序排列，那么那些置顶的便签（也就是 top 值为 1）就会在顶部显示。代码如下：

在字符串集 `public static final String[] PROJECTION_NOTE = new String[]{}`中

添加属性 `NoteColumns.TOP,NoteColumns.STAR,`

定义两个参数：

```
    private String mTop;
    private String mStar;
```

在构造函数中添加获得属性值语句：

```
    mTop = getmTop();
    mStar = getmStar();
```

```
public String getmTop(){return mTop;}
public String getmStar(){return mStar;}
```

在方法 `private void loadFromCursor(Cursor c) {}`中

添加获取 mTop、mStar 值语句：

```
mTop = c.getString(Top_COLUMN);
mStar = c.getString(STAR_COLUMN);
```

完成上述操作，就已经可以获取 mNote 的 top 值了。在 NotesLoginActivity 活动中，方法 private void startAsyncNotesListQuery()是用来同步便签列表并设置排序方式的，这里将排序方式按钮 top 值优先并降序的方式排序，实现置顶：

```
mBackgroundQueryHandler.startQuery(FOLDER_NOTE_LIST_QUERY_TOKEN, null,
        Notes.CONTENT_NOTE_URI, NoteItemData.PROJECTION, selection, new String[] {
                String.valueOf(mCurrentFolderId)
        }, NoteColumns.TOP+ " DESC,"+ NoteColumns.TYPE + " DESC,"
                + NoteColumns.MODIFIED_DATE + " DESC");
```

星标便签不可删除功能，则是在三种不同删除方式进行删除选择便签时跳过那些星标便签（star 值为 1），这里的难点在于，当文件夹内含有便签时，文件夹应该也变成星标便签文件夹，并且不可被删除，所以设置便签星标值时，涉及到调用父便签并修改其 star 属性值，这里就需要通过数据库来进行操作。

在 tool 包内的 DataUtils.java 文件中，方法 batchDeleteNotes()就是用来实现删除选中便签的，传入参数 ids 存放了被选中的待删除便签，只要对 ids 做一些修改即可：

```
for (long id : ids) {
    if(id == Notes.ID_ROOT_FOLDER) {
        Log.e(TAG, "Don't delete system folder root");
        continue;
    }
    /**
     * whether can delete or not
     */
    if (NotesProvider.isStarNote(id) || NotesProvider.isLockedNote(id)){
        continue;
    }
    ContentProviderOperation.Builder builder = ContentProviderOperation
            .newDelete(ContentUris.withAppendedId(Notes.CONTENT_NOTE_URI, id));
    operationList.add(builder.build());
}
```

方法 isStarNote()，就是用来判断是否是星标便签的，是星标便签则跳过，不是则可被删除。此方法是在 Data 包的 NoteProvider.java 文件中实现的：

```
public static boolean isStarNote(Long id) {
    SQLiteDatabase db;
    db = mHelper.getReadableDatabase();
    Cursor cursor = null;
    cursor = db.query(NotesDatabaseHelper.TABLE.NOTE, null, null, null, null, null,
null);
```

```
        //判断游标是否为空
        if (cursor.moveToFirst()) {
            //遍历游标
            for (int index = 1; index < cursor.getCount(); index++) {
                cursor.moveToNext();
                if (cursor.getLong(SqlNote.ID_COLUMN) == id
                        && !cursor.getString(SqlNote.STAR_COLUMN).equals("0")) {
                    cursor.close();
                    return true;
                }
            }
        }
        cursor.close();
        return false;
    }
```

为了使我们清楚地知道便签到底有没有置顶，有没有设成星标，可以在便签列表栏添加图标，用来区分置顶与星标便签以及普通便签：

在 Note_item.xml 中添加布局：

```
<ImageView
    android:id="@+id/iv_top_icon"
    android:layout_width="17dp"
    android:layout_height="17dp"
    android:layout_gravity="top|right"
    android:layout_marginRight="40sp" />

<ImageView
    android:id="@+id/iv_star_icon"
    android:layout_width="17dp"
    android:layout_height="17dp"
    android:layout_gravity="top|right"
    android:layout_marginRight="20sp" />
```

在 NotesListItem.java 文件中通过 bind 方法根据便签 top 与 star 值来设置图标是否显示：

```
if(data.isTOP()){
    mTop.setImageResource(R.drawable.menu_top);
    mTop.setVisibility(View.VISIBLE);
} else{
    mTop.setVisibility((View.GONE));
}
if(data.isSTAR()){
    mStar.setImageResource(R.drawable.star);
    mStar.setVisibility(View.VISIBLE);
} else{
```

```
        mStar.setVisibility(View.GONE);
    }
```

至此，置顶与星标便签功能实现完毕！

- 完善九宫格手势加密

Ui 包：新增 LockPatternView 类用于展示九宫格点；新增活动 SetLockActivity、UnLockActivity、DeletePasscodeActivity 用于设置密码、验证密码与删除密码。

NoteEditActivity 当进行密码设置或者删除密码时，切换到对应的活动界面：

```
//设置密码
case R.id.menu_set_passcode:
    setPasscode();
    break;
//删除密码
case R.id.menu_delete_passcode:
    deletePasscode();
    break;

/**
 * 弹出设置密码界面
 */
private void setPasscode() {
    //确保笔记已写入数据库
    saveNote();
    //初始化一个新的意图，用以跳转至设置密码界面
    Intent intent = new Intent(this,SetLockActivity.class);
    //给意图传入当前便签的 id 参数，便于从设置密码活动中跳转回编辑便签活动
    intent.putExtra(Intent.EXTRA_UID, mWorkingNote.getNoteId());
    startActivity(intent);
    finish();
}

/**
 * 弹出验证密码界面
 */
private void unlockNote(){
    //如果设置了密码，启动输入密码页面
    if (mWorkingNote.hasPasscode() && Locked) {
        saveNote();
        Intent intent = new Intent(this,UnlockActivity.class);
        intent.putExtra(Intent.EXTRA_UID, mWorkingNote.getNoteId());
        startActivityForResult(intent,REQUEST_SET_PASSCODE);
        finish();
```

```
    }
}

/**
 * 删除便签密码
 */
private void deletePasscode() {
    //验证密码
    if (mWorkingNote.hasPasscode()) {
        saveNote();
        Intent intent = new Intent(this,DeletePasscodeActivity.class);
        intent.putExtra(Intent.EXTRA_UID, mWorkingNote.getNoteId());
        startActivity(intent);
        //finish();
        saveNote();
    }
    saveNote();
}
```

当然加密的便签也是不能被删除的，此实现方法与星标便签相同，不再叙述。

以 DeletePasscodeActivity 为例，当完成操作后需要返回到便签编辑界面：

```
public boolean isPassword() {
    if (mPasswordStr.equals(password)) {
        Toast.makeText(DeletePasscodeActivity.this,
                R.string.note_passcode_deleted, Toast.LENGTH_SHORT).show();
        mWorkingNote.setPasscode("");
        mWorkingNote.saveNote();
        /**
        NoteEditActivity Activity = new NoteEditActivity();
        Activity.NoteEditActivity.finish();
        */
        Intent intent = new Intent(DeletePasscodeActivity.this,
                NoteEditActivity.class);
        intent.setAction(Intent.ACTION_VIEW);
        intent.putExtra(Intent.EXTRA_UID, noteId);
        startActivity(intent);
        DeletePasscodeActivity.this.finish();
    } else {
        Toast.makeText(DeletePasscodeActivity.this, "密码不正确",
                Toast.LENGTH_SHORT).show();
    }
    return false;
}
```

在实操过程中发现，上述代码对 Activity 的切换十分混乱，很多时候返回到意料之外的地方(NotesListActivity)，这给用户操作带来了很大的不便。通过阅读分析代码，知道了加密功能的逻辑流程图，引起问题的根本原因在于 back 回退键会调用 destroy 方法，结束掉当前的 Activity，产生一个意外的结束。想要解决这个问题有两个办法，1.监听 back 回退键 2.在创建新的 intent 时考虑到 back 键的意外结束。

1.监听 back 回退键：当回退时回到想要的操作界面，避免混乱：

```
public void onBackPressed() {
    super.onBackPressed();//注释掉这行,back 键不退出 activity
    Intent pre = getIntent();
    //将密码写入数据库
    long noteId = pre.getLongExtra(Intent.EXTRA_UID, 0);
    Intent intent = new Intent(SetLockActivity.this, NoteEditActivity.class);
    intent.setAction(Intent.ACTION_VIEW);
    intent.putExtra("lock",0);
    intent.putExtra(Intent.EXTRA_UID, noteId);
    startActivity(intent);
}
```

建新的 intent 时考虑到 back 键的意外结束：

在 NoteEditActivity 中添加属性：

```
static Activity NoteEditActivity;
```

并在 onCreate 方法中赋值：

```
NoteEditActivity = this;
```

在 DeletePasscodeActivity 活动中创建新的 intent 前添加如下代码：

```
NoteEditActivity Activity = new NoteEditActivity();
Activity.NoteEditActivity.finish();
```

上述两种方法可以解决由于 back 键的调用产生的意外活动结束问题，使得界面操作更具有友好性。

在加密的工程中，发现了一个新的问题，当时间过久之后，用户可能会忘记了密码，又因为加密了的便签是无法删除的，这就会出现部分忘记了密码的加密便签打不开删不去，一直停留在便签列表中，怎么也清理不了，造成了不必要的资源浪费。于是找回密码功能应运而生。

首先在 UnlockActivity 活动的页面中设置一个找回密码的按钮，他的初始状态是不可见的，当密码验证失败后，显示按钮，点击按钮可以找回本条便签的密码：

Activity_lock.xml 的布局如下：

```
<Button
    android:id="@+id/retrieve_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="找回密码"
    android:visibility="gone"/>
```

在 UnlockActivity 中设置 click 监听此按钮：点击按钮出现弹框，弹框内显示密码：

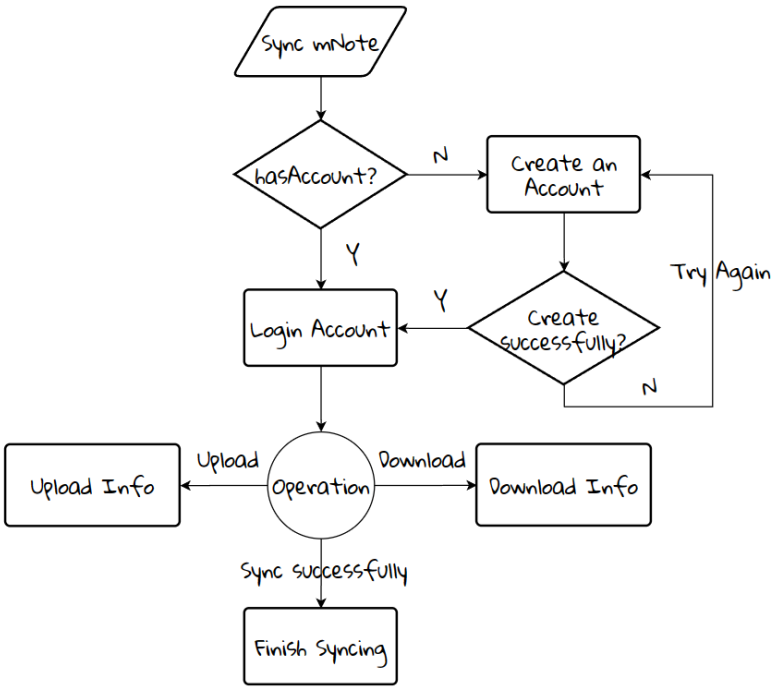
```
public void onClick(View v){
    switch (v.getId()) {
        case R.id.retrieve_password:
            System.out.println("1");
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle(getString(R.string.find_passcode));
            builder.setIcon(android.R.drawable.ic_dialog_alert);
            Intent pre = getIntent();
            final Long noteId = pre.getLongExtra(Intent.EXTRA_UID, 0);
            WorkingNote mWorkingNote = WorkingNote.load(UnlockActivity.this,noteId);
            builder.setMessage(mWorkingNote.getPasscode());
            builder.setPositiveButton(android.R.string.ok, null);
            builder.show();
            break;
        default:
            break;
    }
}
```

这里是通过便签的 Id 索引找到便签的属性，将 Passcode 展示在弹框里。

至此完成九宫格加密功能。

- 云端存储上传与下载

  云端存储是一个相对比较复杂的功能，在编写代码前，先进行逻辑上的分析，最终得到以下代码实现逻辑流程图;



有了流程图，按照流程图的箭头一步步编写代码即可实现功能。

在进行上述操作前，首先要编写一个服务器（连接接口，用于与阿里云数据库 RDS 连接）：

```
/**
 * 此处为您的连接地址、端口、账号、密码等信息。
 */
private static final String url ="jdbc:mysql://ip:3306?" +
              "useSSL=false&ampserverTimezone=GMT";
private static final String user = "account";
private static final String password = "password";

/**
 * 此处为小米便签用户账号 id、用户名、密码、同步数据等信息
 */
private static int user_id;
private static String user_account;
private static String user_pwd;
private static String user_info;
```

以登陆为例，连接 RDS 的代码如下：

```
try {
    /**
     * 连接到阿里云端服务器
     */
    Class.forName("com.mysql.jdbc.Driver");
```

```
        conn = DriverManager.getConnection(url,user,password);
        System.out.println("connect successfully");

        /**
         * 查找用户是否存在
         */
        sql = "select * from mnote.mnote";
        preparedStatement = conn.prepareStatement(sql);
        resultSet = preparedStatement.executeQuery();
        while (resultSet.next()){
            user_id = resultSet.getInt("user_id");
            user_account = resultSet.getString("account");
            user_pwd = resultSet.getString("pwd");
            if (user_account.equals(account) && user_pwd.equals(pwd)){
                return true;
            }
        }
        //捕捉异常。
    } catch (SQLException e) {
        System.out.println("MySQL 操作错误");
        e.printStackTrace();
    }
```

接下来是各个活动功能的实现：

登陆 Login：

```
findViewById(R.id.login).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText editText1 =(EditText) findViewById (R.id.accountName);
        final String outputaccount = editText1.getText().toString();
        EditText editText2 =(EditText) findViewById (R.id.accountPassword);
        final String outputpwd = editText2.getText().toString();
        new Thread(new Runnable() {
            @Override
            public void run() {
                try{
                    if (ConnectRDS.Login(outputaccount,outputpwd)){
                        userAccount = outputaccount;
                        userPwd = outputpwd;
                        Intent intent = new Intent(LoginActivity.this,
                                syncActivity.class);
                        startActivity(intent);
                        finish();
                    } else {
                        showMessage("请检查账号密码！");
```

```
                }
            } catch(Exception e){
                e.printStackTrace();
            }
        }
    }).start();
  }
});
```

点击按钮，创建子线程调用连接接口连接阿里云 RDS。

创建用户 addAccount：

具体代码实现与登陆类似，注意的是，创建用户时，用户名不能重复，否则就会出现混乱，这在接口中可以解决，当创建的用户名已存在则返回重写输入用户名，若不存在则可创建成功：

```
sql = "select * from mnote.mnote";
preparedStatement = conn.prepareStatement(sql);
resultSet = preparedStatement.executeQuery();
while (resultSet.next()){
    if (resultSet.getString("account").equals(newaccount)){
        return "existed";
    }
}
sql = "insert into mnote.mnote(account,pwd) values(?,?)";
preparedStatement = conn.prepareStatement(sql);
preparedStatement.setString(1, newaccount);
preparedStatement.setString(2, pwd);
int result = preparedStatement.executeUpdate();
if (result != 0) {
    System.out.println("注册成功");
    return "success";
}
```

同步 upload & download：

进行 upload 操作前需要将便签数据库里的两个表 note 与 data 封装成 Json 消息包：

```
public static JSONObject uploadnote()throws JSONException {
    JSONObject js = new JSONObject();
    SQLiteDatabase db;
    db = mHelper.getReadableDatabase();
    Cursor cursor = null;
    cursor = db.query(NotesDatabaseHelper.TABLE.NOTE,
            null, null, null, null, null, null);
    //判断游标是否为空
    if (cursor.moveToFirst()) {
        for (int index = 1; index <= cursor.getCount(); index++) {
            JSONObject note = new JSONObject();
```

```
            note.put(NoteColumns.PARENT_ID,cursor.getString(1));
            note.put(NoteColumns.ALERTED_DATE,cursor.getString(2));
            note.put(NoteColumns.BG_COLOR_ID,cursor.getString(3));
            note.put(NoteColumns.CREATED_DATE,cursor.getString(4));
            note.put(NoteColumns.HAS_ATTACHMENT,cursor.getString(5));
            note.put(NoteColumns.MODIFIED_DATE,cursor.getString(6));
            note.put(NoteColumns.NOTES_COUNT,cursor.getString(7));
            note.put(NoteColumns.SNIPPET,cursor.getString(8));
            note.put(NoteColumns.TYPE,cursor.getString(9));
            note.put(NoteColumns.WIDGET_ID,cursor.getString(10));
            note.put(NoteColumns.WIDGET_TYPE,cursor.getString(11));
            note.put(NoteColumns.SYNC_ID,cursor.getString(12));
            note.put(NoteColumns.LOCAL_MODIFIED,cursor.getString(13));
            note.put(NoteColumns.ORIGIN_PARENT_ID,cursor.getString(14));
            note.put(NoteColumns.GTASK_ID,cursor.getString(15));
            note.put(NoteColumns.VERSION,cursor.getString(16));
            note.put(NoteColumns.TOP,cursor.getString(17));
            note.put(NoteColumns.STAR,cursor.getString(18));
            note.put(NoteColumns.PASSCODE,cursor.getString(19));
            js.put(cursor.getString(0),note);
            if(!cursor.isLast()){
                cursor.moveToNext();
            }
        }
    }
    System.out.println(js);
    return js;
}
public static JSONObject uploaddata()throws JSONException {
    JSONObject js = new JSONObject();
    SQLiteDatabase db;
    db = mHelper.getReadableDatabase();
    Cursor cursor = null;
    cursor = db.query(NotesDatabaseHelper.TABLE.DATA,
            null, null, null, null, null, null);
    //判断游标是否为空
    if (cursor.moveToFirst()) {
        for (int index = 1; index <= cursor.getCount(); index++) {

            JSONObject data = new JSONObject();
            data.put(DataColumns.MIME_TYPE,cursor.getString(1));
            data.put(DataColumns.NOTE_ID,cursor.getString(2));
            data.put(DataColumns.CREATED_DATE,cursor.getString(3));
            data.put(DataColumns.MODIFIED_DATE,cursor.getString(4));
```

```
            data.put(DataColumns.CONTENT,cursor.getString(5));
            data.put(DataColumns.DATA1,cursor.getString(6));
            data.put(DataColumns.DATA2,cursor.getString(7));
            data.put(DataColumns.DATA3,cursor.getString(8));
            data.put(DataColumns.DATA4,cursor.getString(9));
            data.put(DataColumns.DATA5,cursor.getString(10));
            js.put(cursor.getString(0),data);
            if(!cursor.isLast()){
                cursor.moveToNext();
            }
        }
    }
    System.out.println(js);
    return js;
}
```

随后将得到的 json 消息包上传到云端数据库，上传的方法与前者相同。

进行 download 操作时，抓取云端数据库的 json 消息包，转到本地后进行解析，对 database 进行处理，主要的操作时数据库的插入修改：

```
public static void syncTABLE(JSONObject JSONInfo) throws JSONException {
    SQLiteDatabase db;
    db = mHelper.getReadableDatabase();
    String sql;
    JSONObject NOTE = null;
    JSONObject DATA = null;
    try {
        NOTE = (JSONObject) JSONInfo.get("NOTE");
        DATA = (JSONObject) JSONInfo.get("DATA");
    } catch (Exception e) {
        e.printStackTrace();
    }

    sql = "DELETE FROM" + " " + TABLE.NOTE;
    db.execSQL(sql);

    assert NOTE != null;
    Iterator<String> NOTEKeys = NOTE.keys();
    while (NOTEKeys.hasNext()){
        String NOTEKey = NOTEKeys.next();
        JSONObject NOTEInfo = (JSONObject) NOTE.get(NOTEKey);
        sql = "INSERT INTO note(_id,parent_id,alert_date,bg_color_id,created_date,"
+
                "has_attachment,modified_date," +
                "notes_count,snippet,type,widget_id,widget_type," +
                "sync_id,local_modified,origin_parent_id,gtask_id," +
```

```
            "version,top,star,passcode) values(?,?,?,?,?,?,?,?,?," +
            "?,?,?,?,?,?,?,?,?,?,?)";
    Object bindArgs[] = new Object[] {
            NOTEKey,
            NOTEInfo.getString("parent_id"),
            NOTEInfo.getString("alert_date"),
            NOTEInfo.getString("bg_color_id"),
            NOTEInfo.getString("created_date"),
            NOTEInfo.getString("has_attachment"),
            NOTEInfo.getString("modified_date"),
            "0",//NOTEInfo.getString("notes_count")
            NOTEInfo.getString("snippet"),
            NOTEInfo.getString("type"),
            NOTEInfo.getString("widget_id"),
            NOTEInfo.getString("widget_type"),
            NOTEInfo.getString("sync_id"),
            NOTEInfo.getString("local_modified"),
            NOTEInfo.getString("origin_parent_id"),
            NOTEInfo.getString("gtask_id"),
            NOTEInfo.getString("version"),
            NOTEInfo.getString("top"),
            NOTEInfo.getString("star"),
            NOTEInfo.getString("passcode")
    };
    db.execSQL(sql,bindArgs);
}
assert DATA != null;
Iterator<String> DATAKeys = DATA.keys();
while (DATAKeys.hasNext()){
    String DATAKey = DATAKeys.next();
    JSONObject DATAInfo = (JSONObject) DATA.get(DATAKey);
    sql = "INSERT INTO data(_id,mime_type,note_id,created_date," +
            "modified_date,content,data3,data4,data5)" +
            "values (?,?,?,?,?,?,?,?,?)";
    Object bindArgs[] = new Object[] {
            DATAKey,
            DATAInfo.getString("mime_type"),
            DATAInfo.getString("note_id"),
            DATAInfo.getString("created_date"),
            DATAInfo.getString("modified_date"),
            DATAInfo.getString("content"),
            DATAInfo.getString("data3"),
            DATAInfo.getString("data4"),
            DATAInfo.getString("data5")
```

```
        };
        db.execSQL(sql,bindArgs);
    }
}
```

当设计方案十分明确（如流程图逻辑性强，可行性高）时，编写代码的难度并不大，重点还是在需求分析。到这儿所有的功能都已经实现完毕。

3.4 sonarqube 维护

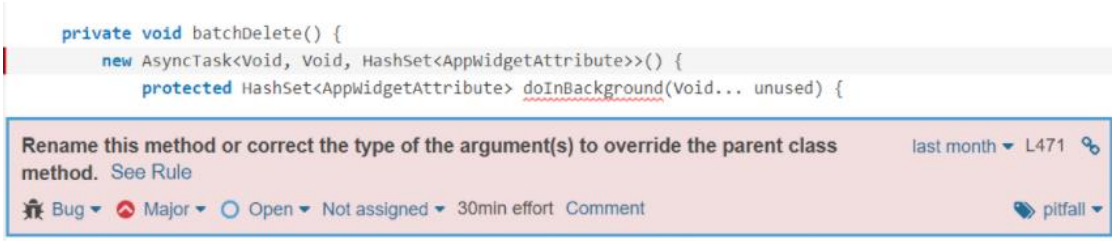
1. ui 包

A. 创建的资源未正常释放

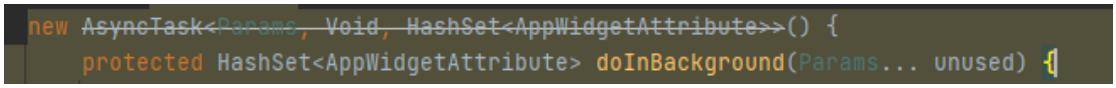
解决方法：try 语句后面跟随括号可以管理释放资源，将两语句放在 try 语句后括号内即可解决。

```
try (InputStreamReader isr = new InputStreamReader(in);BufferedReader br = new BufferedReader(isr)){
    in = getResources().openRawResource(R.raw.introduction);   //加载软件自带的第一条便签内容
    char [] buf = new char[1024];
    int len = 0;
    while ((len = br.read(buf)) > 0) {
        sb.append(buf, 0, len);
    }
} catch (IOException e) {
```

B. 函数与父类由于参数发生了冲突

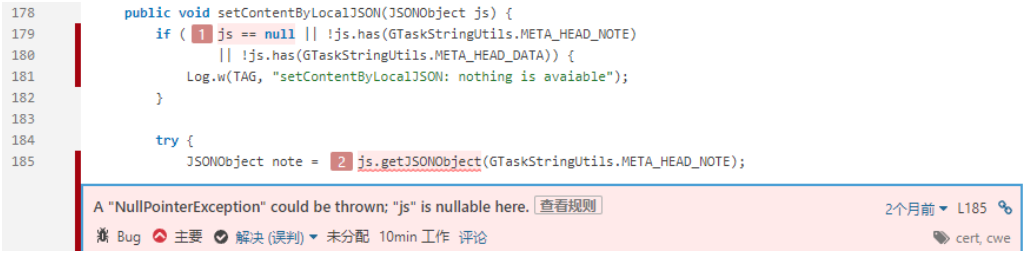


解决方法：将参数改为一致并在定义类时声明此参数



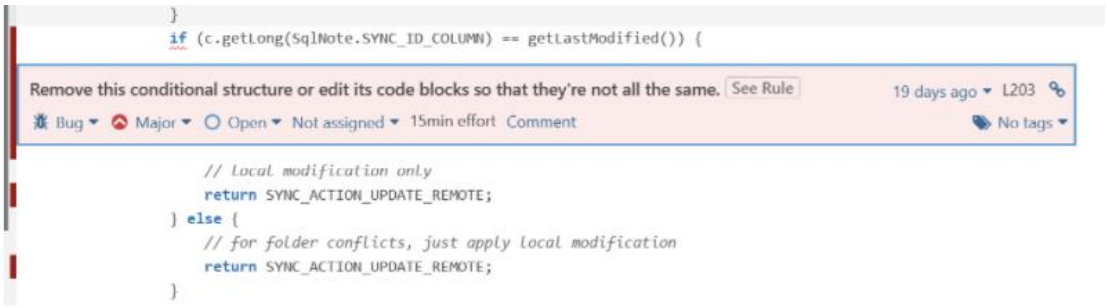
2. gtask 包

A. "js"是可以为空的，可以抛出"NullPointerException"

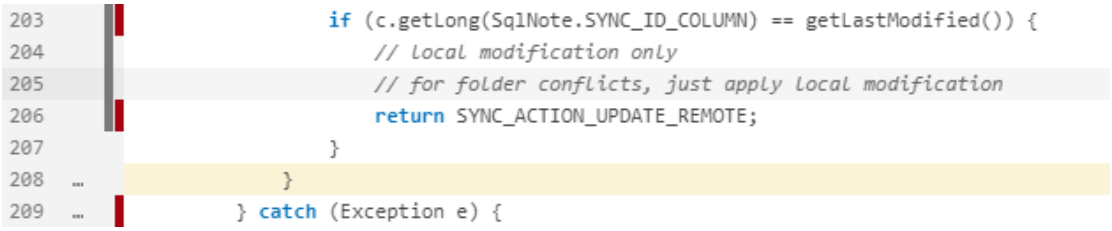


解决方法：if 语句已经进行了空值判断，js 在 try 语句中不会为空，此处应为 sonarqube 误报。

B. if，else 语句逻辑相同



解决方法：将 else 语句删除



C. 常量强制转换

```
public boolean login(Activity activity) {
    // we suppose that the cookie would expire after 5 minutes
    // then we need to re-login
    final long interval = 1000 * 60 * 5;
```

Cast one of the operands of this multiplication operation to a "long". See Rule 19 days ago L115

Bug Minor Open Not assigned 5min effort Comment based-on-misra, cert, cwe, overflow, sa...

解决方法：将计算数值改为 long 类型

```
115 …        final long interval = 1000L * 60L * 5L;
```

D. 缺少空值检测

```
while (iter.hasNext()) {
    Map.Entry<String, Node> entry = iter.next();
    node = entry.getValue();
1   doContentSync(Node.SYNC_ACTION_ADD_LOCAL, node, null);
```

"NullPointerException" will be thrown when invoking method "doContentSync()". See Rule 20 days ago L334

Bug Major Open Not assigned 10min effort Comment cert, cwe

```
            break;
        case Node.SYNC_ACTION_DEL_LOCAL:
            meta = mMetaHashMap.get( 2  3  c.getString(SqlNote.GTASK_ID_COLUMN));
            if (meta != null) {
                GTaskClient.getInstance().deleteNode(meta);
```

解决方法：添加 if 判断 c 是否为空，若为空值则直接抛出异常

```
492             case Node.SYNC_ACTION_DEL_LOCAL:
493 …               if (c == null){
494                     Log.e(TAG, "error");
495                     return;
496                 }
497 …               meta = mMetaHashMap.get(c.getString(SqlNote.GTASK_ID_COLUMN));
```

3. tool 包

需要在 finally 块中将 FileOutputSttream 关闭

解决方法：添加 finally 代码块将 FileOutputSttream 关闭

```
300 …            FileOutputStream fos = null;
301 …            try {
302 …                fos = new FileOutputStream(file);
303 …                ps = new PrintStream(fos);
304              } catch (FileNotFoundException e) {
305                  e.printStackTrace();
306                  return null;
307              } catch (NullPointerException e) {
308                  e.printStackTrace();
309                  return null;
310 …            } finally {//鍗抽櫎FileOutputStream
311                  if(fos != null){
312                      try{
313                          fos.close();
314 …                    }
315 …                  catch(IOException e){
316                          e.printStackTrace();
317                      }
318                  }
319              }
320 …            return ps;
321          }
322      }
```

4. 维护后的开源软件

类别	定性和定量描述
SonarQube 对开源软件的分析情况	12 bug
标注的代码行数量	约 1000 行
新增的代码量	约 1700 行（不含.xml 文件）
新增代码中重用的代码量	新增 7bug
新增代码中独立编写的代码量	约 1200 行
SonarQube 对维护后开源软件的分析情况	Bug 完全消除

功能	实现者	代码行数	总计
修改APP图标	赵升	1行	约 60行
统计输入字符	赵升	约 60行	
载入欢迎界面	肖锟钰	约 60行	约260行
更换背景图片	肖锟钰	约200行	
置顶重要便签	马坚	约100行	约650行
设置星标便签	马坚	约200行	
便签手势加密	马坚	约350行	
云端上传下载	葛宏波、马坚	约700行	约700行